User Heap, GDI Heap, Wind32 Heap, Menu32 Heap, GDI32 Heap. Then set the Level to 2. Now execute the Stress program.

2. With Stress running, start your product from its icon.
3. Exercise all of your product's functions, and make sure they are stable even when the system is under stress.
4. Switch video modes "on the fly," and make sure your product is still operational.
5. Test printing and all other input/output functions supported.
6. Close Stress, and then start one popular 16-bit application and one popular 32-bit application to see whether your product will still run without problems while these are active.

ON WINDOWS NT

1. Start the Performance Monitor (Perfmon.exe).
2. Choose to view counters from "Memory," "Objects," and any other object-types that are relevant to your application.
3. Run your application for extended periods of time, exercising all of your product's functions, and ensure that all systematic changes in system resources are accounted for.
4. Switch video modes "on the fly," and make sure your product is still operational (not supported on Windows NT prior to version 4.0).

4.8. OLE/DCOM

Required:

☑ The application must be an OLE 2.0 container or object server. Although it can be both a server and a container, it does not have to be.

Tip: If you want to host Java™ or ActiveX controls in your application, you should choose to be an OLE container.

OBJECT SERVER REQUIREMENTS:

Required:

☑ The user must be able to drag the object to any container, including other internal documents, the desktop, or documents in another OLE-supporting application.

Required:

☑ Object servers must pass basic OLE 2.0 server functionality testing. You can test this by dragging an object created by your product into an application such as Microsoft Word.

Recommended:
Object servers should close automatically after delivering their object.

CONTAINER REQUIREMENTS

Required:
☑ Your product must pass basic OLE 2.0 container functionality testing. You can test this by creating an object in any standard OLE server, such as Microsoft Word, and dragging it into your product.

Required:
☑ Containers must provide an adequate Object command on the Insert menu.

Required:
☑ The application must implement the IDropTarget() and/or IDropSource() OLE interfaces for drag-and-drop functionality. This means that the application can accept drops from any IDropSource() or can drag onto any IDropTarget(), or both.

Tip: Many applications get Drag and Drop for free from their edit controls and from RichEdit. If you use the Windows 95 and Windows NT common controls, or the RichEdit control for your edit controls, your application already has support for Drag and Drop.

Recommended:
Support ActiveX Accessibility, including the WM_GETOBJECT, IDispatch(), and IAccessible() interfaces. For more information, see 7.3.11, ActiveX Accessibility.

Recommended:
Provide support for modifier keys.

Recommended:
Provide linking and in-place activation.

Recommended:
Containers should support the OLE-structured storage compound file format. Summary information should be usable and complete as presented by the 32-bit Microsoft Windows operating system shell. The fields are as follows:

Author	Comments
Create Time/Date	Keywords
Last Printed	Last Saved By
Last Saved Time/Date	Name of Creating Application
Number of Characters	Number of Pages
Number of Words	Revision Number
Security	Templates
Thumbnail	Total Editing Time

Recommended:
Applications should expose all functionality through OLE Automation (generally, the IDispatch() interface). This facilitates scripting your applications from scripting languages.

4.9. Communications: TAPI 2.0

Required:
☑ Communications applications that dial, answer, and otherwise control telephone calls on equipment such as modems, ISDN adapters, and telephones must control those calls using the Windows Telephony API (TAPI).

Required:
☑ Telephony *enabled* applications must, at minimum, use the Assisted Telephony interface and pass requests for dialing of calls to a telephony *centric* application such as *Phone Dialer* that uses the full TAPI interface for call control. If the application performs more than simple outbound dialing, it must implement the full TAPI interface and meet the requirements for telephony *centric* applications. When using the Assisted Telephony interface, the following additional behaviors are recommended.

Recommended:
In telephony *enabled* applications, phone number entry fields for country code and area code should use the Dialing Properties current location settings as defaults. These defaults can be obtained by calling **tapiGetLocationInfo**.

Recommended:
A telephony *enabled* application should pass the number dialed to **tapiRequestMakeCall** in canonical/international form. This is done so that the call manager application placing the call can properly apply the user's Dialing Properties settings. This is not necessary if the number dialed is an internal extension or the user has otherwise explicitly instructed the application to dial the digits "as is" without applying Dialing Properties.

Recommended:
The parameters passed to **tapiRequestMakeCall** by a telephony *enabled* application should also include the application name, and, if available, the name of the person being called.

Required:

☑ Telephony *centric* applications must use the full Windows Telephony API (TAPI) for telephony-related functions. They must not control telephony-related equipment through proprietary interfaces, drivers, or APIs other than TAPI, unless a Windows Telephony service provider (TSP) is not available for such equipment; however, the Logo will not be granted to an application that controls telephony hardware directly if that application is published by the same company that produces such telephony hardware (instead, the company must also provide a telephony service provider to permit the hardware to be controlled through TAPI). In particular, the application must not access a modem by directly opening the serial port using CreateFile ("COMx"), although a "Direct to COMx" mode may be available for accessing non-modem devices. Telephony *centric* applications must also meet the following requirements.

Required:

☑ Telephony *centric* applications must apply the user's Dialing Properties settings to numbers to be dialed. This can be accomplished by passing the number to the **lineTranslateAddress** function before the call is placed. This number must be in canonical/international form unless the number to be dialed is an internal extension or the user has otherwise explicitly instructed the application to dial the digits "as is" without applying Dialing Properties (they should pass to **lineTranslateAddress** so that tone/pulse dialing settings can be applied).

Required:

☑ Telephony *centric* applications must allow the user to select the device to use for calls. The device name can be obtained from the **LINEDEVCAPS** structure returned by the TAPI **lineGetDevCaps** function.

Required:

☑ If a telephony *centric* application is performing extended control functions on a serial device that is also accessible through TAPI (such as a fax application issuing Class 1 or 2 fax AT commands to a fax modem), it must use **LINEBEARERMODE_PASSTHROUGH** as defined in TAPI rather than opening the serial device directly, so that the device is properly shared with other TAPI apps when not in use. The application must not open the serial device directly using **CreateFile**.

Recommended:

In addition to the above requirements, the following behaviors are strongly recommended for all telephony *centric* applications:

- User entry of phone numbers should be in separate fields for country code, area/city code, and local number. The country code and area code fields should default to the user's current location country code and area code obtained from **lineGetTranslateCaps** function. The

country code selected may consist of a drop-down list of countries obtained from the **lineGetCountry** function.

- The application should provide user access to Dialing Properties via a button, menu, or context-menu selection that invokes the **lineTranslateDialog** function. If a button is used, it should be located close to wherever phone numbers are entered. If a phone number has already been entered by the user when the button or menu item is clicked, the entered phone number should be passed to **lineTranslateDialog** in canonical/international form, so that the display of actual digits to be dialed and control over dialing of the call as a long distance call will work properly.
- The application should support calling card dialing in stages when necessary (for example, the underlying equipment does not support certain dialing control codes). Such support includes handling LINEERR_DIAL* error codes from **lineMakeCall** and **lineDial**, dividing the dialable string into segments, passing each segment individually to **lineDial**, and prompting the user between segments to listen for the expected tone or signal. This process is described in detail in the TAPI section of the Win32 SDK documentation.
- The application should properly respond to **LINEDEVSTATE_REINIT** and **PHONESTATE_REINIT** messages by shutting down the application's use of TAPI at the earliest opportunity so that TAPI can be restarted.
- The application should properly handle the **LINE_CREATE** message so that telephony devices that are added by the user while the application is active become immediately available for use.
- If the application handles inbound calls, it should allow the user to set whether or not the application is the highest-priority (preferred) application for each type of call (media mode) that the application handles, using the **lineSetAppPriority** function.
- If the application handles inbound calls, it should respect the toll-saver features of other applications by calling **lineGetNumRings** to determine the ring to answer on, before answering any inbound call.
- The TAPI line device ID on which the call is intended to be dialed should be passed as a parameter to **lineTranslateAddress**, so that future implementation of line-specific Dialing Properties will correctly function.
- If the application is a "call manager" application intended to be the primary dialing application for a user, it should function as an Assisted Telephony Server application by calling **lineRegisterRequestRecipient**, processing **LINE_REQUEST** messages, calling **lineGetRequest**, and making the requested calls. It should also allow the user to select whether or not the application is the highest priority application for handling such requests by calling **lineSetAppPriority**.

Required:

☑ If the application uses TAPI 2.0 functions, it must either explicitly link to TAPI32.dll (by calling GetProcAddress) or be linked to theTAPI32.lib *explicit* link library so that it will load on both Windows 95 (TAPI 1.4) and Windows NT 4.0 (TAPI 2.0). The application should gracefully degrade its features so that it works as well as possible within the limitations of TAPI 1.4 when running on Windows 95.

4.10. Cross-Platform

Required:

☑ You must provide a version of your application which runs on both Windows NT and Windows 95. By definition, Intel processor support is a minimum requirement, since Windows 95 is a single-platform operating system. You are not required to develop multiple versions of your application to run on multiple processor platforms.

Required:

☑ Windows NT is a multiple-platform operating system. It runs on Intel, MIPS, Alpha, and PowerPC processors. If you create versions of your application to run natively on each of these processors, each version must be tested separately for your application to qualify to use the Designed for Windows NT and Windows 95 Logo.

4.11. UNC/LFN Support

Required:

☑ Your application must support the Universal Naming Convention (UNC). UNC paths allow logical connections to network devices without the need to specifically reference a network drive letter. Your application does not need to be network-aware per se, but it does need to work seamlessly in a network environment. The system must be able to locate the network server and path with the UNC name even over a modem connection. Supporting UNC does *not* mean that the application is disallowed from presenting network drive letters to users. It merely means that the user must have the option of using only the UNC path name.

Required:

☑ All file operations in file-based applications must support UNC paths without the need for specifying drive letters.

Verification:
To test direct network browsing with UNC paths: Open a file, and use the **Save As** command to save it with a Long File Name (LFN) and UNC paths (for example, \\ServerName\MySubdirectory\MyLongFileName) to a standard server. It should be possible to save and retrieve files without specifically referencing a network drive letter.

Note: An LFN is 260 characters, which in general includes 3 bytes for "**<driveletter>:**\", 255 bytes for the filename+extension, and 2 bytes for the null terminator. A UNC path has 2 bytes for "\\" instead of 3 bytes for "**<driveletter>:**\", and that the path may not include an extension (filetype).

Required:
If your application saves files that are exposed to the user, your application must support LFNs with all of the following required features:

- Users must be able to enter names of 255 characters, including all uppercase and lowercase standard characters, embedded spaces, embedded periods, etc.
- Leading and trailing spaces must be stripped by the **Save As** command. To test this, enter a name with leading and trailing spaces, such as "####This is a test###", retrieve the file, and then make sure the spaces are deleted.

Note: Here and throughout this handbook, the number sign (#) indicates a spacebar space.

- Question marks anywhere in the file name must prevent the file from being saved. No error message needs to be displayed.
- The characters within the quotation marks " . + , ; = [ ] " must be supported anywhere in the name, including leading and trailing. These should not cause any error conditions.

Verification:
Here is how to test whether your product properly handles LFNs. Basically, LFNs must do the following:

- Allow periods, plus signs, commas, semicolons, equal signs, and square brackets anywhere.
- Not save leading or trailing spaces. (You can test for this by removing the name inserted into the Save As dialog box and typing "###test###" or similar text. The program should strip the spaces and add an extension, returning the file name "test.ext".)
- Not save question marks.
- Support 255 characters (including the path and extension).
- Save to a UNC path, such as \\Server\Directory\Filename.

You should test each of the allowed file names in the following list. When your application saves each file, it should add an extension and save it to the hard disk.For example, "test." will save as "test..ext".

test

....test....

....####test####....

....####te..st####....

test....

....test

.test

test.

test#test#test#test

test#1234567890[on through 260]

You should also test the following list of file names, which save to the hard disk as indicated:

test (saved as "test.ext")

###test (saved as "test.ext")

test### (saved as "test.ext")

###..test..##(saved as "..test...ext")

test#;#+#,#=#[#] (saved as "test#;#+#,#=#[#].ext")

####....####test.test####....## (saved as "....####test.test####.....ext")

\\folder#one\folder#two\folder#three\folder#four\file

Required:

☑ The application must use LFNs for displaying all documents and data files in the shell, in title bars, in dialog boxes and controls, and with icons.

Exception:

Certain development tools as well as applications that have a valid need to save files without default file extensions may be exempt from the requirement to support trailing periods. All variations must be detailed in your Vendor Questionnaire.

Required:

☑ If the application supports automatic launching to a data file when clicking on data file icons in Windows Explorer, these icons must launch to the LFNs and not display the 8.3 short name.

Recommended:

Hiding the .xxx extension names in the application itself is strongly recommended, but not required.

Applications are allowed to use "labeling schemes" in which a user is saving, for example, a report type or a game state, without actually creating a file that is exposed to the user in Windows Explorer. Consider the following:

- Games often use a labeling scheme, as do financial packages and databases, which generate different kinds of reports that are not actually data files on their own. An example would be a game that allows the user to save 10 different game states with up to 20-letter names. As long as those names do not appear in Windows Explorer, this situation is okay.
- If users think they are saving a real file name, the situation is different. If you save a game or multimedia state to a file name, and the file name is fully exposed in Windows Explorer, it must be a fully supported LFN.

Required:

☑ You must test your LFN functionality on Windows NT FAT, NTFS, and compressed NTFS partitions.

5. Exceptions, Exemptions, And Additional Requirements

5.1. Non-file-based Applications

The following points define a file-based application and a non-file-based application:

- A non-file-based application is one that is *not* primarily used to create, edit, and save files (although file operations may be common ancillary tasks).
  - A somewhat less direct example is the following: An investment-analysis application might allow a user to perform a query to determine a common stock's current price, then analyze the stock's historical price/performance by creating various descriptive statistics, charts, or graphs, which the user might save. While a file may be created and saved in this latter example, editing files is not the primary purpose of the application. Rather, it is a simple convenience provided by the application. This is a non-file-based application.
  - Many accounting packages and Personal Information Manager (PIM) products are considered non-file-based applications. If you're uncertain whether your product is considered non-file-based, please explain your case in the Vendor Questionnaire.
  - Generally, applications that run exclusively in full-screen mode are not, for Logo purposes, considered file-based. An application that runs exclusively in full-screen mode is one that cannot, for example, be windowed or resized. Usually, this refers to applications without Minimize or Maximize buttons, or those that use the full-screen APIs or a full-screen bitmap for aesthetic/design purposes. However, these will be considered on a case-by-case basis.
- File-based applications have as their primary purpose the creation and editing of documents and include **Open**, **Save**, and **Close** commands, typically on the **File** menu of the application.
  - The primary purpose of applications such as Microsoft Word and Microsoft Excel is to allow the user to create, edit, and manipulate files. Therefore, they do need to support the requirements for file-based applications in order to be eligible to license the Designed for Windows NT and Windows 95.
  - Utilities are considered non-file-based applications.

Exception:
If your product is a non-file-based application, the requirements to support OLE and UNC pathnames do not apply. A direct example of this might be a utility, multimedia reference title, a game, or applications like Terminal and Clock, which for the most part do not save files in any way and, if they do, it's done as a facility for saving user profiles, for example.

5.2. Development Tools

Development tools are products such as languages and compilers that create executable files or interpreted code modules that mimic the action of executable files when used with a run-time engine of the developing application. These products must meet all the general requirements, with the following additions and exceptions.

Exception:
OLE drag-and-drop support is not required within the tool's design environment.

Required:
☑ If Windows is one of the compiler's or development tool's target platforms, then the product must be capable of generating applications that can meet all the Designed for Windows NT and Windows 95 Logo requirements.

Exception:
Development tools which create Java applications do not need to meet the above requirement.

Exception:
Certain "interactive entertainment authoring tools" are not required to create applications that support OLE and UNC pathnames, as follows:

- When the interactive entertainment authoring tools are intended only for authoring non-file-based multimedia applications.
- Developers of this kind of tool must submit a sample application.
- Developers of this kind of tool must advise their customers that if they create a file-based application using this tool (whether or not they use other tools), in order for that application to be eligible for the Designed for Windows NT and Windows 95 Logo, it must meet the file-based requirements, including OLE support.

Required:
☑ The development tool must provide an easy point-and-shoot way (commonly known as wizards or experts) to create applications with OLE container or object support, or provide this functionality by default.

Required:

Development tools are required to submit a sample application in uncompiled form for testing. When compiled, this application must demonstrate the following:

- The Win32 PE format.
- Large and Small Icons for its executable file.
- The ability to save/retrieve files with LFNs.
- The ability to save/retrieve files with UNC file names.
- OLE Container and/or Server drag and drop functionality.

Exception:

Sample applications do not need to meet any of the other Windows Logo requirements, such as install/uninstall, etc.

Recommended:

Development tools should provide the ability to create applications that use multiple threads.

5.3. Utilities

Utilities are products such as shell extensions, disk optimizers, antivirus software, certain query engines, and accessibility aids for people with disabilities. They must meet all the general requirements contained in this document with some additions and exceptions.

Required:

In order for a utility to receive the Logo, it must include in the same package (box) meaningful functionality for *both* Windows 95 and Windows NT environments.

Exception:

Products such as disk utilities that implement operating-system-specific functionality are exempt from those requirements that do not make sense on the other platform. For example, a set of utilities might include a disk defragmenter which operated on Windows NT but, not on Windows 95. If so, the product must include significant elements which are useful on Windows 95. A detailed explanation of Windows NT vs. Windows 95 differences must be provided in the Vendor Questionnaire.

Exception:

Certain components of utilities may be 16-bit, such as those that must use the Exclusive Volume Locking API, soft interrupts, or components that must talk directly to 16-bit drivers. The user interface and other components of these applications must be 32-bit and use the Windows 95 thunking mechanism to access these 16-bit components.

Required:

☑ Products that use the Exclusive Volume Locking API to access the 8.3 aliases directly and manipulate them, such as disk optimizers, file utilities, and antivirus software, must be tested extensively to ensure that these products function properly and do not damage the file allocation table (FAT) or the LFN structure. To test:

Verification:

Run the utility (that is, your product), and manipulate several LFN files. Open these files with their LFN and with their 8.3 aliases, and make sure they still relate to the same file.

Check overall LFN file structure to make sure its integrity has not been damaged.

5.4. Games and Multimedia Applications

Required:

☑ All products that use DirectX must fail gracefully on Windows NT 3.51, but function properly on Windows NT 4.0 and later versions.

Required:

☑ All products that use Direct3D APIs must fail gracefully on Windows NT 3.51 and Windows NT 4.0. Failing gracefully when using OS-specific APIs such as Direct 3D does not mean the application is exempt from providing as complete functionality as possible on both operating systems. For example, games must still function reasonably on both Windows NT and Windows 95--regardless of what API sets are used--in order to qualify for the Logo. Alternative ways of providing comparable functionality (such as using 2D DirectDraw™ calls on Windows NT instead of Direct3D) must be used to provide comparable user experience on both operating systems.

5.5. Add-On Products

Add-on products are accessory products, such as wizards, templates, and macros, which are not executable files. If a wizard, template, or macro is always packaged *only* with the 32-bit product it supports, it is considered part of that product for Logo testing. This section applies only to add-on products which are packaged and marketed *separately* from the product they support.

Exception:

An add-on product does not have to be an executable (.exe).

Required:

☑ To qualify for the Logo, add-ons must be used by a 32-bit product that is capable of earning the Designed for Windows NT and Windows 95 Logo.

Required:

☑ If the separately marketed add-on products are also included in a suite of applications, they must be tested separately and with the suite for the suite to qualify for the Designed for Windows NT and Windows 95 Logo.

Required:

☑ Add-on products must follow all other Logo requirements that apply to any functionality they provide.

Examples:

- Any executable code in the add-on products must be 32-bit code.
- Add-on products must provide automated installers and uninstallers that follow the user interface rules as defined in UI/Shell.
- Add-on products that use native data file types must register them and their icons.
- Add-on products that use the mouse or keyboard must support the Accessibility requirements.

6. Frequently Asked Questions (FAQ)

Q: Are there new Logo requirements in this version?

A: Yes, there are. Highlights of the new requirements include:

- Accessibility requirements and recommendations
- Reformat of document to clarify requirements
- Expanded Install and Uninstall sections
- Added Internet requirements
- Equal emphasis on Windows NT and Windows 95
- Clarified Stability and Functionality requirements
- Added TAPI 2.0 requirements
- Added Cross-Platform requirements

Q: If my product(s) already earned the Designed for Windows 95 Logo, do I need to retest them for the new Logo?

A: No, the revisions do not affect already logod products.

Q: If I just submitted a product to testing, will it automatically be tested with the revised requirements for the new Designed for Windows NT and Windows 95 Logo?

A: No. The new Logo and new requirements are available beginning July, 1996. You may request your product be tested for either the old Designed for Windows 95 Logo or the new Logo.

Q: How important is the information in the Vendor Questionnaire?

A: *Very!* The information in the Vendor Questionnaire on the disk is used in a number of ways that benefit you.

- The marketing/artwork contact is the person who will receive the countersigned Logo license agreement and the artwork. Please make sure that this information is correct.
- Marketing contact information is used to send information to you about co-marketing programs or general status communiqués from the marketing team.
- Customer service information, the product description, sales support information, and the product category are used on the Web site (http://www.microsoft.com/windows/thirdparty/) where Microsoft lists products that bear the Designed for Windows NT and Windows 95 Logo.
- The primary contact is the person whom VeriTest needs to contact regarding general technical issues or questions.

- The technical contact is the person whom VeriTest contacts for more specific product questions.
- Development details are used for VeriTest's testing process to determine special-case or special-appeal potential.

Q: How long will the process take?

A: Once VeriTest has received your complete submission, Logo testing will require eight business days at VeriTest, followed by five business days at Microsoft.

Q. Is there any way to speed up the process?

A: Potentially. There are ways to ensure that there won't be a delay in the process and that your testing can be done quickly. Before submitting your product to VeriTest, we advise you to do the following:

- Read this Logo Handbook for suggestions on creating Logo-compliant applications.
- Test your application thoroughly on both Windows NT and Windows 95.
- Make sure all the pertinent information is filled out in the Vendor Questionnaire.
- List as the main contact the person to whom we should send the license and artwork kit.
- Include the instructions on how to use your product.

Q. How do I get the artwork now, so I can include it in my packaging and long-lead advertising?

A: Per FTC regulations, we cannot provide you with the artwork for the Logo until you are licensed to use the Logo. If you want to use stickers on your box, there is an order form for Logo stickers (enclosed with the paper format of the Pretest Kit, and to come soon on http://www.microsoft.com/windows/thirdparty/winlogo/). You can place your orders in advance; however, they will not be filled until you receive the license to use the Logo.

Q. Who is VeriTest?

A: VeriTest has been in the business of quality assurance and compatibility testing since 1987 and has served many large and small companies in the computer industry. It is a private company, completely independent of any other industry entity. You may wish to visit VeriTest's Web page at http://www.veritest.com for more information.

You do not have to schedule your test with VeriTest, but you must first obtain a Pretest Kit from Microsoft (see Section II, "Obtaining the Logo"), and send VeriTest all the required elements before it will begin testing. You may also be able to perform other Microsoft Logo tests at VeriTest simultaneously.

Under the contract you sign with VeriTest, it has eight business days to return a result to you, starting from the day after it receives *all* elements of your application package. To date, VeriTest has met this deadline for all of the hundreds of products already submitted to its lab and is committed to returning your results on time or sooner. VeriTest understands the tight deadlines you may be working under and is willing to help in whatever way possible.

VeriTest is also available to answer questions during the Logo-testing process. Please contact them at logolab@veritest.com.

Q: Can I get interim results? How is testing done?

A: No. Testing is conducted using a standardized protocol with freshly installed versions of Windows 95 and Windows NT Workstation on computers that are referenced to a known configuration. Because of the considerable setup time, once testing of your product begins, it must continue to completion. If testing is halted for any reason, it must be restarted from the beginning on a newly standardized system. For this reason, although VeriTest is available for consultation once your test begins, it cannot release "partial results."

Q: How are results sent?

A: Here is a schedule of how the results and any retesting are handled:

VeriTest will send a fax to your primary contact as soon as it has obtained a result.

- A follow-up hard copy of the result is also mailed through U.S. mail.
- If you have not requested confidentiality, your test results, license agreement, and Vendor Questionnaire are then forwarded through an overnight service to Microsoft.
- If you have questions or concerns about your test results, you may contact VeriTest by phone to discuss them.
- If you need to make changes and submit your product for a retest, you must sign a Retest Agreement and send it, with your revised build, to VeriTest.
- Retests are given the highest priority in the testing queue.

Q: How complete does my product need to be to submit it for testing?

A: Due to early packaging deadlines, we will not always be receiving the final, shrink-wrappable version of your software. However, the product must be in "final beta," that is, *fully installable using your installer, fully functional,* and nearly ready for shipping. Once the product has been tested, you should not be adding new functionality that affects the Logo criteria. If functionality is incomplete, you will fail the Logo test. (After your product is final, you will need to send a copy of it both to VeriTest and to Microsoft. Each company will conduct random retests of licensed products to ensure that final products still meet the Logo requirements.)

Q: If I already had my suite of applications tested, but I plan to sell my bundled applications separately, do these need to be tested separately?

A: Yes. These are handled as retests for install/uninstall and Registry issues. The reverse is also true: If VeriTest checked your products independently, it must retest your bundled installation program when you deliver the products as a suite. Please see the Step-by-Step instructions for pricing at http://www.veritest.com.

Q: Can a sub-product or add-on product qualify for the Designed for Windows NT and Windows 95 Logo if the master product does not?

A: Perhaps. It may be okay if a sub-product, that is, one that depends on another product for its data creation, is the only one that qualifies for the Logo. This will come up, for example, with a product such as a "front end" for a remote Microsoft SQL Server™ database. However, these judgments are made on a case-by-case basis. A product that requires a 16-bit product for significant portions of its own engine's execution obviously would not qualify.

Q: Can VeriTest test a product that requires beta third-party products to function?

A: Yes. VeriTest can test with beta versions of third-party products on which the primary product depends, even if those third-party products do not yet have the Logo themselves; for example, third-party DLLs may be used for testing. However, overall stability is crucial for the Logo test.

Q: If I already had my application tested on the Intel platform, but I also have MIPS, Alpha, and PowerPC versions, do these need to be tested separately?

A: Yes. These are handled as retests for install/uninstall issues. For the non-Intel versions of your application to qualify for Logo use, each version must be tested on its native platform. Please see the instructions for pricing at http://www.veritest.com.

Q: Do add-on modules to my Logo-bearing product also need to be tested? I have an accounting program that has add-on modules for different tasks. When those are published, do I need to send them to VeriTest?

A: Yes. All add-on modules need to be tested if you want them to carry the Logo.

Q: If I'm a general 32-bit application developer, where should I look for development information?

A: Developer information can be found on the Microsoft developer web site: http://www.microsoft.com/win32dev.

Q: If I'm a game developer, where should I look for information on developing Windows 32-bit games?

A: Game developer information can be found on the Microsoft developer web site: http://www.microsoft.com/gamesdev.

Q: Is the Windows compatible Logo still available for 16-bit products?

A: No. Microsoft no longer licenses the Windows compatible Logo.

Q: Where can I get more information and updates on the Designed for Windows NT and Windows 95 Logo program?

A:

- Keep up with the Designed for Windows NT and Windows 95 Logo home page at: http://www.microsoft.com/windows/thirdparty/winlogo/. This should be your primary source for up-to-date information.
- VeriTest, Inc., the independent laboratory that performs testing for the program, also maintains a helpful information page at: http://www.veritest.com. Look under "Microsoft Logo Programs."
- Updated versions of the Logo handbook will be published on Microsoft Developer's Network (MSDN). For information on how to subscribe to MSDN, see http://www.microsoft.com/msdn.
- The Logo handbook is also published as part of the Windows NT Workstation Resource Kit (Version 4.0 and higher).
- Read Microsoft Press® publications:
  - The Windows Interface Guidelines for Software Design
  - Programmer's Guide to Microsoft Windows 95
- Contact the Windows Logo Department with questions by sending an e-mail message to winlogo@microsoft.com.
- To contact the Logo Laboratory at VeriTest, send an e-mail message to logolab@veritest.com or send a fax to the Designed for Windows NT and Windows 95 Logo Lab at VeriTest at (310) 399-1760.

Q: I understand that Microsoft also offers logos for Designed for Windows 95, Office Compatible, and the Designed for BackOffice programs. How can I get information on these?

"Designed for Windows 95"
Vendors may continue to apply for and obtain this Logo until January 1, 1997. Because the "Designed for Windows NT and Windows 95" Logo is a superset of the Windows 95 requirements, passing the Windows test automatically qualifies the vendor to use the Windows 95 Logo as well. However, if you only wish to support the Windows 95 Logo, you may do so until the January 1, 1997, deadline. To obtain information about this Logo on the World Wide Web (WWW), please see:

- http://www.microsoft.com/windows/thirdparty/winlogo/
- You may also wish to consult http://www.veritest.com

Microsoft Office Compatible

This is a popular Logo for those who want end users to know that their application follows the look and functionality and integrates closely with Microsoft Office Standard and Professional. The user interface criteria for 16-bit and 32-bit products is the same, but 32-bit products must also obtain the Designed for Windows NT and Windows 95 Logo before they can qualify for the Office Compatible program. VeriTest, Inc. also administers this testing program under the authority of Microsoft, and discounts are available for multiple-Logo tests submitted simultaneously.

- To get information on the World Wide Web: http://www.microsoft.com/msoffice/ofccomp/
- To contact VeriTest concerning the Office Compatible Logo, send an e-mail message to officelogo@veritest.com, or send a fax to (310) 399-1760.

Designed for Microsoft BackOffice

The Microsoft BackOffice integrated suite of server applications for enterprise computing systems includes the Windows NT Server network operating system, the Microsoft SQL Server database server, Microsoft SNA Server, Microsoft Systems Management Server, and Microsoft Mail Server.

To get the BackOffice Pretest Kit, send an e-mail message to bckoffc@microsoft.com, or send a fax request to (206) 936-5458. You may also send mail to BackOffice Logo Department, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399.

To get general information on the BackOffice Logo criteria, contact one of the following:

- Contact the Microsoft Developer Solutions Phone/Fax service. Dial (206) 635-2222. Relevant documents are numbered in the 840–859 range.
- On the World Wide Web (http://www.microsoft.com/backoffice/), go to the Partner Forum, Designed for Microsoft BackOffice Logo Program.
- On MSN, open Categories, Computers and Software, Computer Companies and Organizations, Software Companies, Microsoft, BackOffice, Partner Forum. Or, Go msbackoffice.

7. Appendixes

7.1. Key Windows 95/Windows NT Workstation Architectural and API differences

The application programming interface for Windows 95 and Windows NT is the same--the Win32 API. However, Windows 95 and Windows NT are different operating systems with different capabilities. There is technology present in Windows NT that is not present in Windows 95, and vice versa. As a result, some of the functions that make up the API are not implemented on both of the platforms.

This appendix provides an overview of those differences by comparing Windows NT version 4.0 with the retail release (in August, 1995) of Windows 95. Subsequent updates to Windows 95 and Windows NT will remove some of these differences and introduce new ones.

This appendix does not claim to be complete, although it does cover the majority of areas that will be of interest to application developers. Differences in the device driver architecture are not detailed here. This appendix provides information only at the function-call level of granularity. Functions that are supported on both platforms, but have some parameters that are meaningful only on one platform, are not detailed here. For complete information on programmatic differences between Windows 95 and Windows NT, consult the API documentation in the most recent Win32 SDK.

WINDOWS 95 ONLY

7.1.1. Direct3D is not available on Windows NT.

DirectX is not available on Windows NT 3.51. DirectDraw, DirectPlay™, DirectSound™, and DirectInput™ are provided as redistributable DLLs for Windows 95, and are built into the operating system for Windows NT version 4.0. Direct3D is also available as a redistributable for Windows 95, but Direct 3D is not available for Windows NT version 4.0.

Functionality	Windows 95	Windows NT 3.51	Windows NT 4.0
DirectDraw	provided as redistributable	not available	built into operating system
DirectPlay	provided as redistributable	not available	built into operating system
DirectSound	provided as redistributable	not available	built into operating system, runs in emulation mode only
DirectInput	provided as redistributable	not available	built into operating system
Direct3D	provided as redistributable	not available	not available

7.1.2. Independent color matching is not supported on Windows NT.

CheckColorsInGamut	ColorMatchToTarget
CreateColorSpace	DeleteColorSpace
EnumICMProfiles	FreeImageColorMatcher
GetColorSpace	GetDeviceGammaRamp
GetICMProfile	GetLogColorSpace
LoadImageColorMatcher	SetColorSpace
SetDeviceGammaRamp	SetICMMode
SetICMProfile	UpdateICMRegKey

7.1.3. Plug and play is not supported on Windows NT.

7.1.4. Flat thunks are not supported on Windows NT.

7.1.5. Pen for Windows support is not available on Windows NT.

WINDOWS NT ONLY

7.1.6. Unicode is not fully supported on Windows 95.

On Windows NT, all functions that accept strings as parameters have both an ANSI and a wide-character version. The functions are post-fixed by "A" and "W," that is, TextOutA accepts an ANSI string and TextOutW accepts a Unicode string. The wide-character version of the Win32 API functions are generally not supported on Windows 95. The notable exceptions (functions that are supported on Windows 95) are:

MultiByteToWideChar	WideCharToMultiByte
EnumResourceLanguages	EnumResourceNames
EnumResourceTypes	ExtTextOut
FindResource	FindResourceEx
GetCharWidth	GetCommandLine

GetTextExtentExPoint
GetTextExtentPoint
MessageBoxEx
TextOut
GetTextExtentPoint32
lstrlen
MessageBox

7.1.7. Windows NT security is not supported on Windows 95.

AccessCheck
AccessCheckAndAuditAlarm
AddAccessAllowedAce
AddAccessDeniedAce
AddAce
AddAuditAccessAce
AdjustTokenGroups
AdjustTokenPrivileges
AllocateAndInitializeSid
AllocateLocallyUniqueId
AreAllAccessesGranted
AreAnyAccessesGranted
CopySid
CreatePrivateObjectSecurity
DdeImpersonateClient
DeleteAce
DestroyPrivateObjectSecurity
DuplicateToken
EqualPrefixSid
EqualSid
FindFirstFreeAce
FreeSid
GetAce
GetAclInformation
GetFileSecurity
GetKernelObjectSecurity
GetLengthSid
GetPrivateObjectSecurity
GetProcessWindowStation
GetSecurityDescriptorControl
GetSecurityDescriptorDacl
GetSecurityDescriptorGroup
GetSecurityDescriptorLength
GetSecurityDescriptorOwner
GetSecurityDescriptorSacl
GetSidIdentifierAuthority
GetSidLengthRequired
GetSidSubAuthority
GetSidSubAuthorityCount
GetTokenInformation
GetUserObjectSecurity
ImpersonateNamedPipeClient
ImpersonateSelf
InitializeAcl
InitializeSecurityDescriptor
InitializeSid
IsValidAcl
IsValidSecurityDescriptor
IsValidSid
LookupAccountName
LookupAccountSid
LookupPrivilegeDisplayName
LookupPrivilegeName
LookupPrivilegeValue
MakeAbsoluteSD
MakeSelfRelativeSD
MapGenericMask
ObjectCloseAuditAlarm
ObjectOpenAuditAlarm
ObjectPrivilegeAuditAlarm
OpenProcessToken
OpenThreadToken
PrivilegeCheck
PrivilegedServiceAuditAlarm
RevertToSelf
SetAclInformation
SetFileSecurity
SetKernelObjectSecurity
SetPrivateObjectSecurity
SetSecurityDescriptorDacl
SetSecurityDescriptorGroup
SetSecurityDescriptorOwner
SetSecurityDescriptorSacl
SetThreadToken
SetTokenInformation
SetUserObjectSecurity

7.1.8. Event logging is not supported on Windows 95.

BackupEventLog
ClearEventLog
CloseEventLog
DeregisterEventSource
GetNumberOfEventLogRecords
GetOldestEventLogRecord
NotifyChangeEventLog
OpenBackupEventLog
OpenEventLog
ReadEventLog
RegisterEventSource
ReportEvent

7.1.9. Service control manager is not supported on Windows 95.

ChangeServiceConfig
CloseServiceHandle
ControlService
CreateService
DeleteService
EnumDependentServices
EnumServicesStatus
GetServiceDisplayName
GetServiceKeyName
LockServiceDatabase
NotifyBootConfigStatus
OpenSCManager
OpenService
QueryServiceConfig
QueryServiceLockStatus
QueryServiceObjectSecurity
QueryServiceStatus
RegisterServiceCtrlHandler
ServiceMain
SetServiceBits
SetServiceObjectSecurity
SetServiceStatus
StartService
StartServiceCtrlDispatcher
UnlockServiceDatabase

7.1.10. Multiple desktops are not supported on Windows 95.

CloseDesktop
CreateDesktop
EnumDesktops
EnumDesktopWindows
OpenDesktop
OpenInputDesktop
SwitchDesktop
GetDesktopWindow
GetThreadDesktop
SetThreadDesktop

7.1.11. Server side named pipes are not supported on Windows 95.

It does not make sense to list functions in this instance, since the named pipe functions can be used for either server or client side. Only the server-side functionality does not work on Windows 95.

7.1.12. Dynamic updating of the resources in portable executable (PE) files is not supported on Windows 95.

BeginUpdateResource
UpdateResource
EndUpdateResource

7.1.13. I/O completion ports are not supported on Windows 95.

CreateIoCompletionPort
GetQueuedCompletionStatus
PostQueuedCompletionStatus

7.1.14. Server-oriented socket calls are not supported Windows 95.

TransmitFile
AcceptEx
GetAcceptExSockaddrs

7.1.15. Certain advanced GDI functions are not supported on Windows 95.

7.1.15.1. World transforms

SetWorldTransform
GetWorldTransform
ModifyWorldTransform
CombineTransform

7.1.15.2. Advanced blt functions

MaskBlt
PlgBlt

7.1.15.3. Printer forms

AddForm
DeleteForm
EnumForms
GetForm
SetForm

7.1.15.4. Printer change notification

FindClosePrinterChangeNotification
FindFirstPrinterChangeNotification
FindNextPrinterChangeNotification
WaitForPrinterChange

7.1.16. Cryptography is not supported on Windows 95.

(This feature is supported on Windows NT 4.0 only.)

CryptAcquireContext
CryptCreateHash
CryptDecrypt
CryptDeriveKey
CryptDestroyHash
CryptDestroyKey
CryptEncrypt
CryptExportKey
CryptGenKey
CryptGenRandom
CryptGetHashParam
CryptGetKeyParam
CryptGetProvParam
CryptGetUserKey
CryptHashData
CryptHashSessionKey
CryptImportKey
CryptReleaseContext
CryptSetHashParam
CryptSetKeyParam
CryptSetProvider
CryptSetProvParam
CryptSignHash
CryptVerifySignature

7.1.17. Fibers are not supported on Windows 95.

(This feature is supported on Windows NT 4.0 only.)

ConvertThreadToFiber
CreateFiber
DeleteFiber
GetCurrentFiber
GetFiberData
SwitchToFiber